
# HDFS

# HADOOP DISTRIBUTED FILE SYSTEM

# Agenda

- ✓ HFDS Design
- ✓ HDFS Architecture
- ✓ HDFS Storage Daemons
- ✓ Name Node, Data Node
- ✓ Secondary Name Node & Check-pointing
- ✓ HDFS Storage Architecture
- ✓ HDFS File Write & Read
- ✓ Hadoop High Availability
- ✓ Basic HDFS Operations

# HDFS Design

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.

**Very Large Files**

100s of GBs or even TBs.

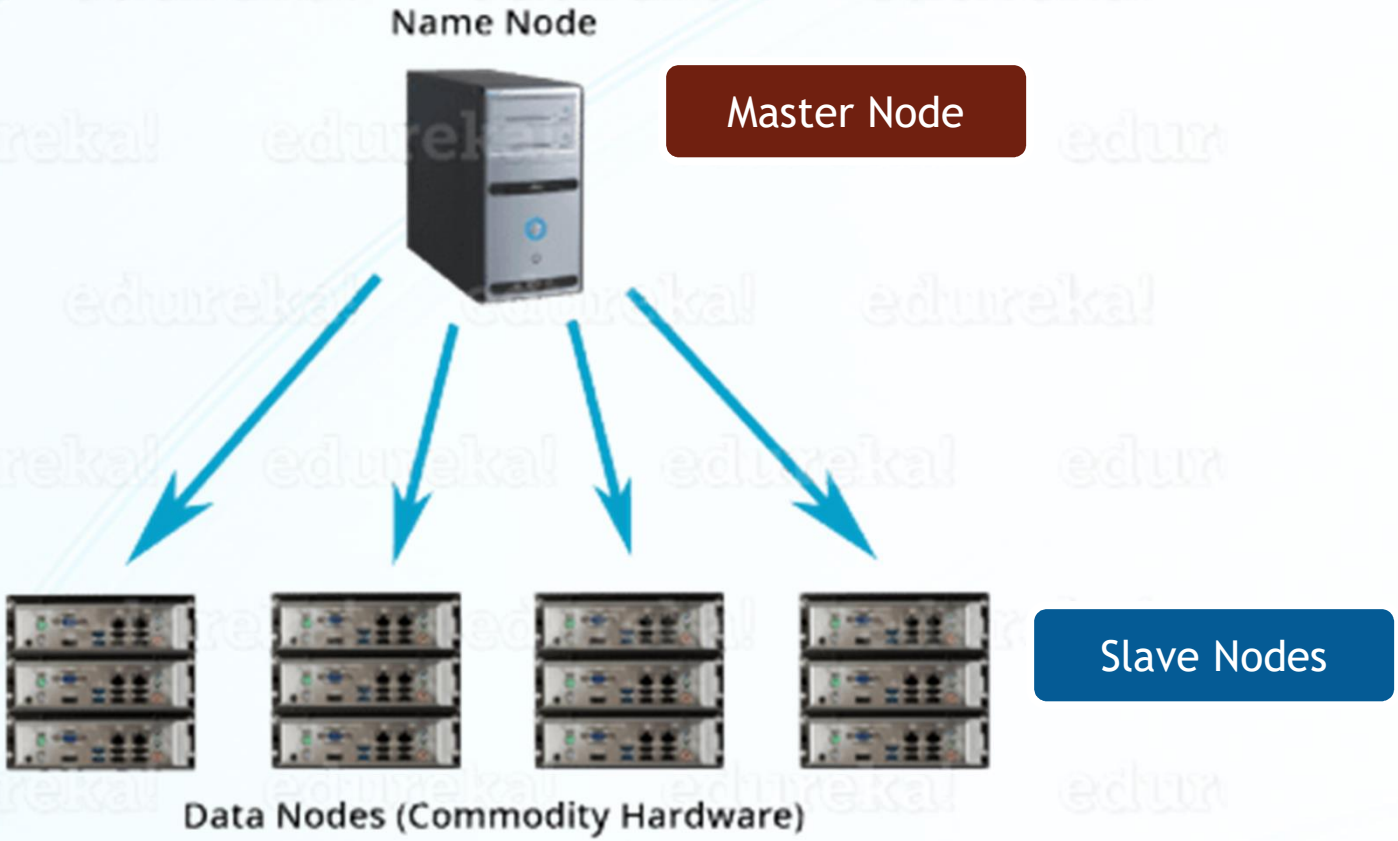
**Streaming Data Access**

Built for "Write-once Ready-many-times" pattern. Time to read the whole dataset is more important than the latency in reading the first record.

**Commodity Hardware**

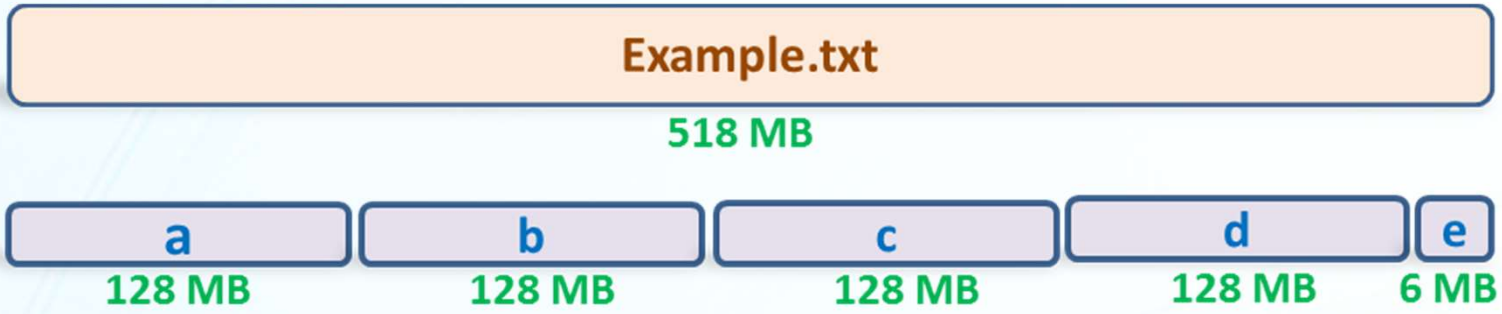
Where chance of node failure is high.

# Hadoop Master/Slave Architecture

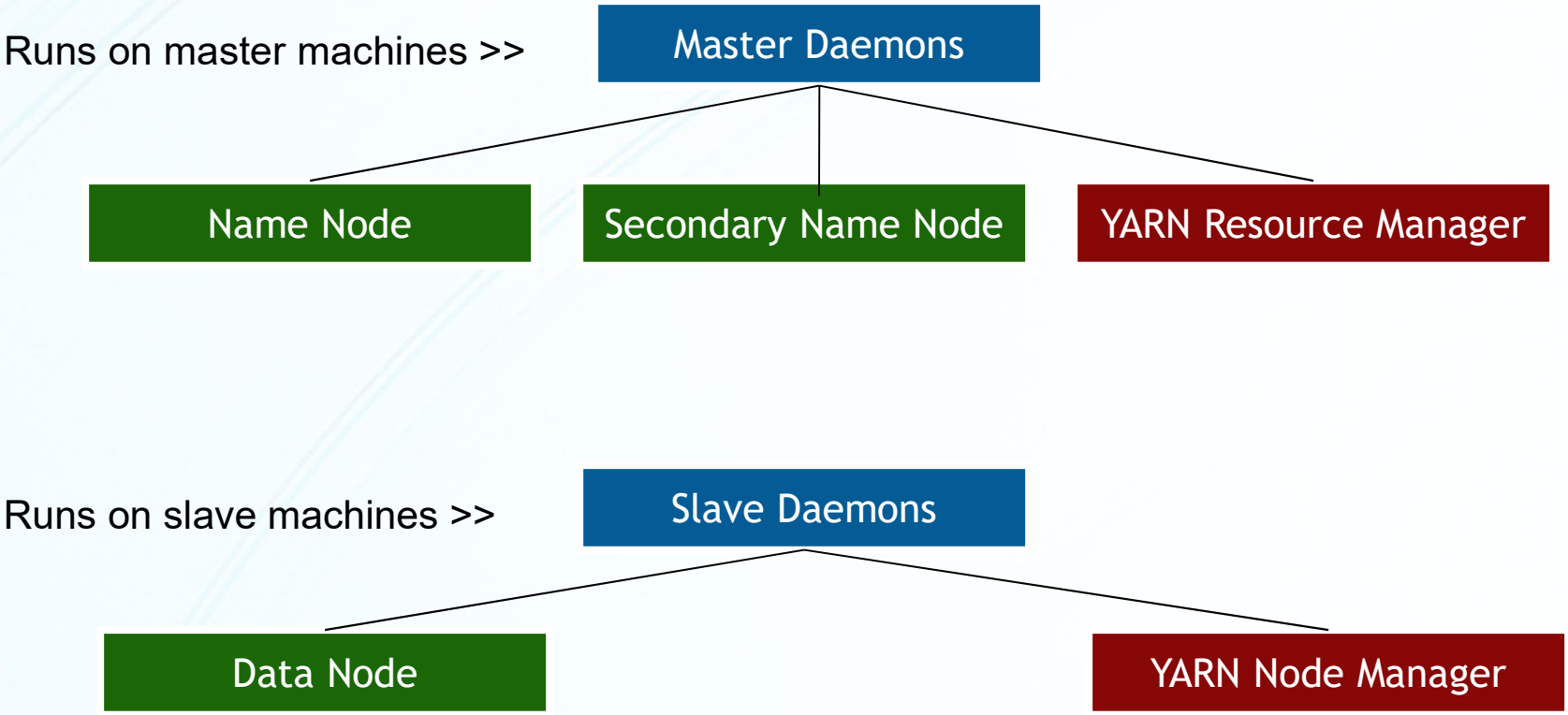


# Blocks

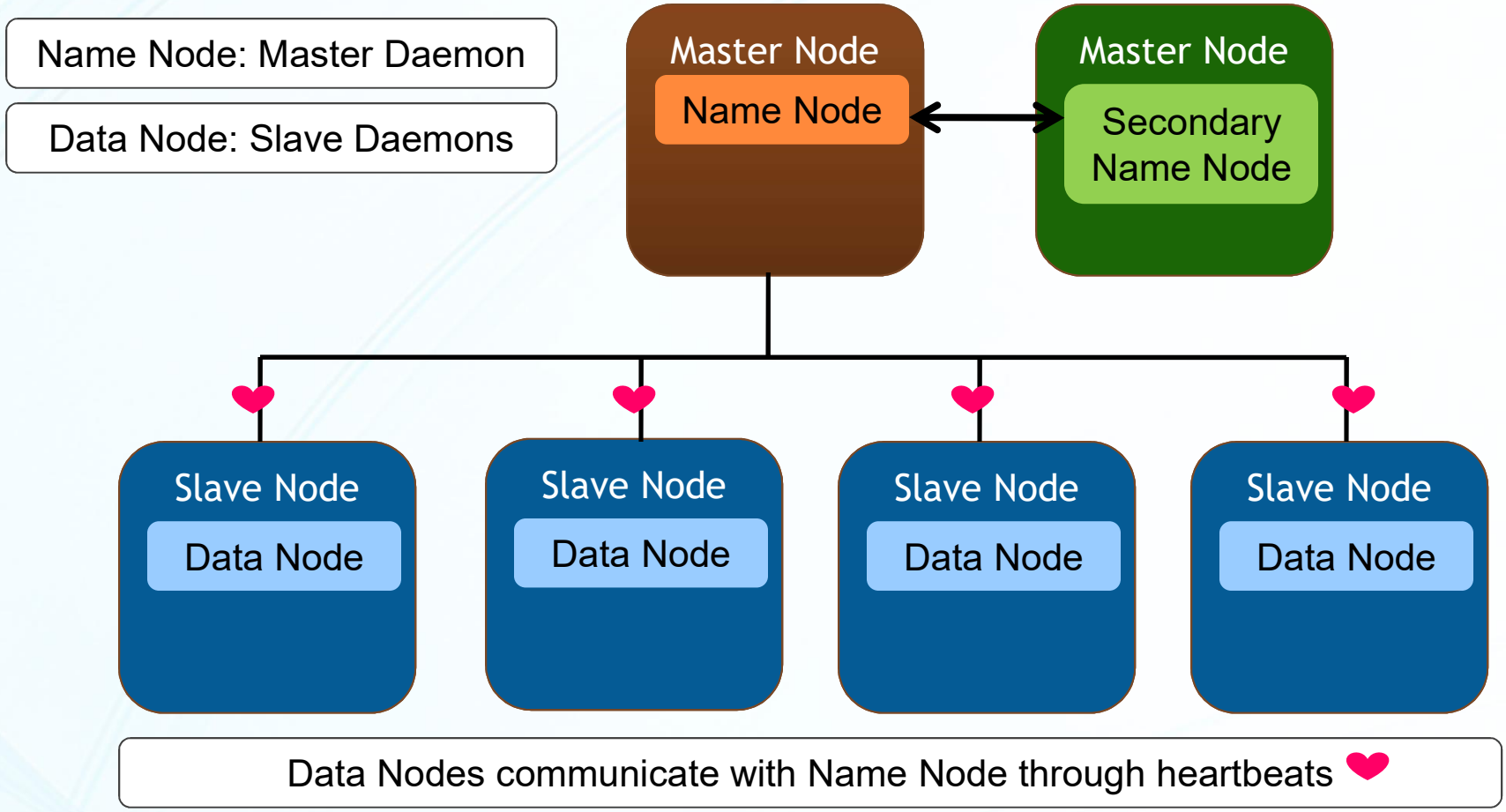
- In HDFS, files are broken into blocks of 128MB and are distributed across various nodes in the cluster.
- Blocks are the basic unit of abstraction in HDFS
- **Advantages:**
  - a file can be larger than any single disk in the network
  - simplifies the storage subsystem
  - fits well with replication for providing fault tolerance and availability



# Hadoop Daemon Processes

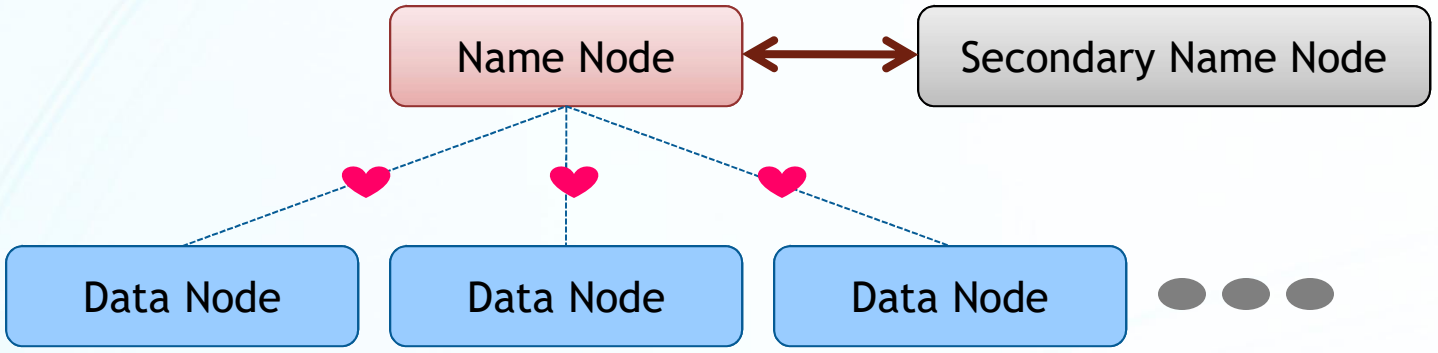


# HDFS Storage Daemons



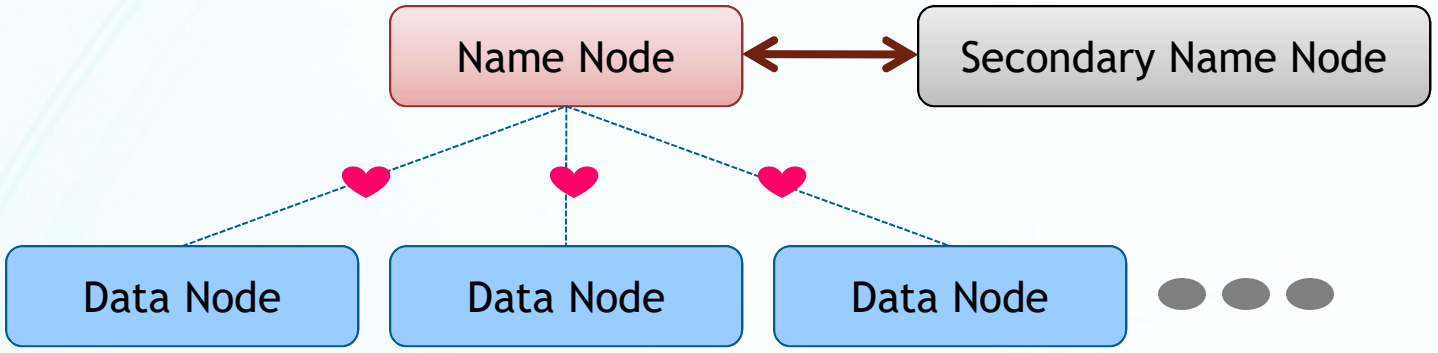
# Name Node

- Name Node is the master daemon that maintains and manages Data Nodes
- Maintains meta data about data blocks such as location of blocks stored, size of the files, permissions, hierarchy etc in a file called **FsImage**.
- Receives heart-beat and block report from all Data Nodes.



# Data Node

- These are slave daemons that runs on each slave machines
- Stores and manages actual data
- Serves Read and Write requests from the client
- Sends heartbeat to Name Node once every 3 sec.



# FsImage & Edit Log Files

## FsImage File

- FSImage file contains the complete state of the file system namespace since the start of the NameNode upto the last checkpoint.
- Each node is an internal representation of a file or directory's metadata and contains such information as the file's replication level, modification and access times, access permissions, block size, and the blocks a file is made up of.
- For directories, the modification time, permissions, and quota metadata is stored.

## EditLog File

- EditLog file is stored in-memory and records all incremental changes happening on HDFS and is periodically merged with FsImage during check-pointing process.

# Secondary Name Node

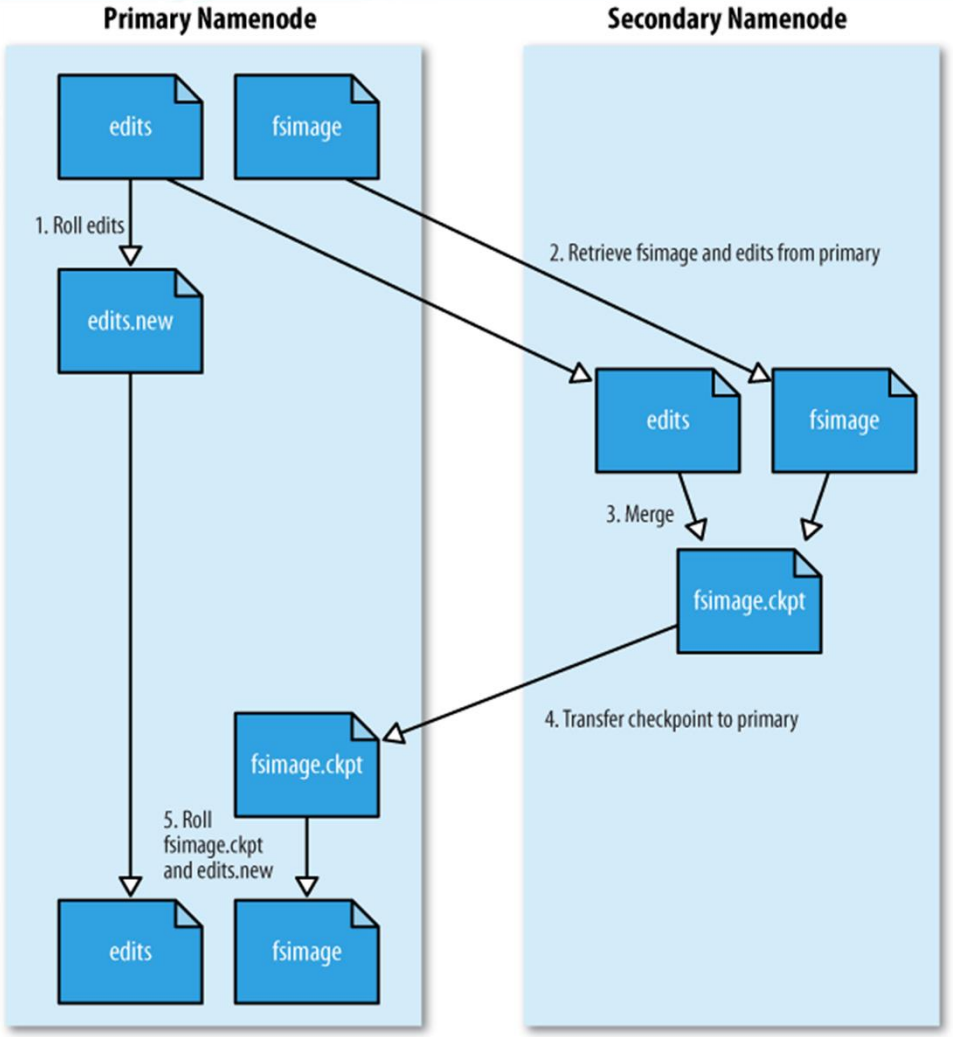
**Check-pointing** is the process of combining edit-logs with FsImage

SNN takes the responsibility of check-pointing, there by reducing load on Name Node

Allows faster fail-over as it prevents the edit-log from getting too huge.

Check-pointing happens periodically.

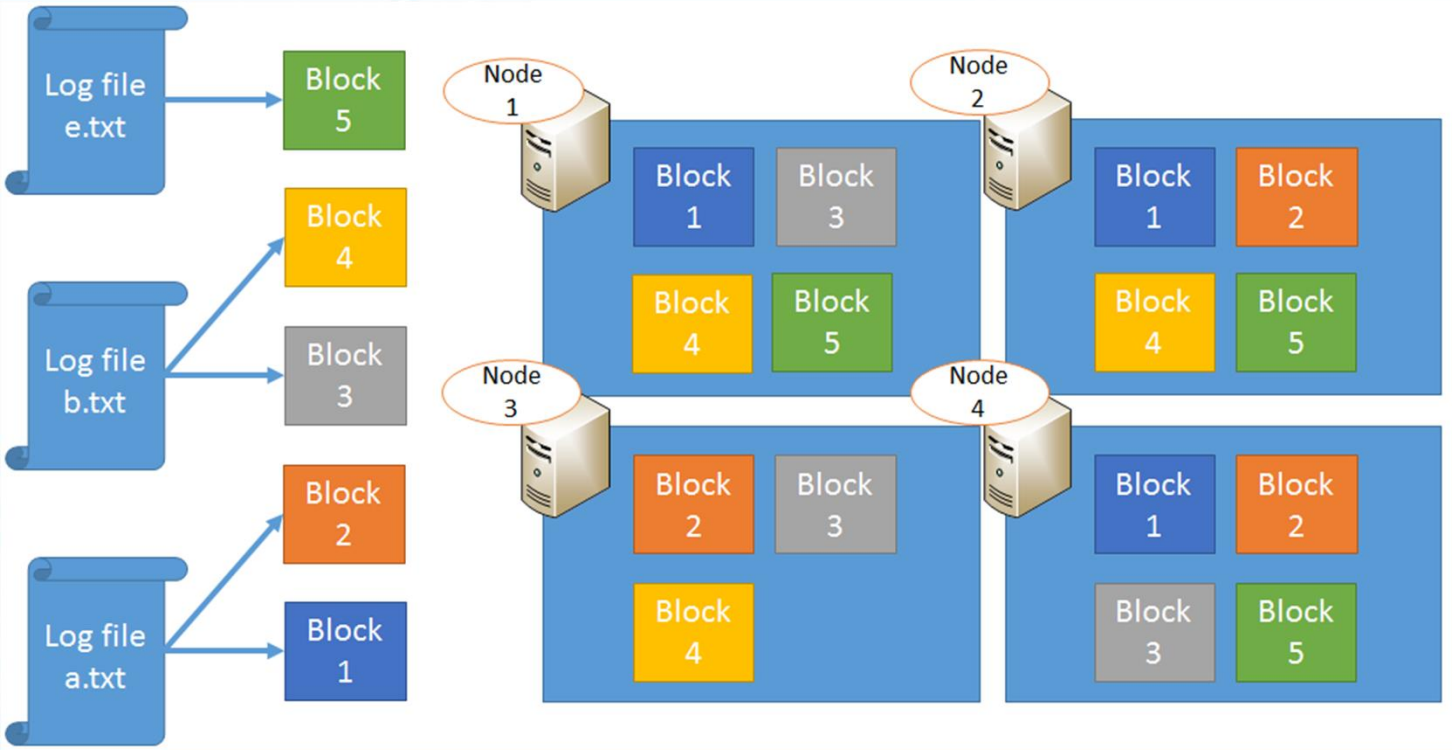
SNN is not a hot stand-by for NN. It just does metadata backup and is used to rebuild a failed NN



# How is data stored in Data Nodes?

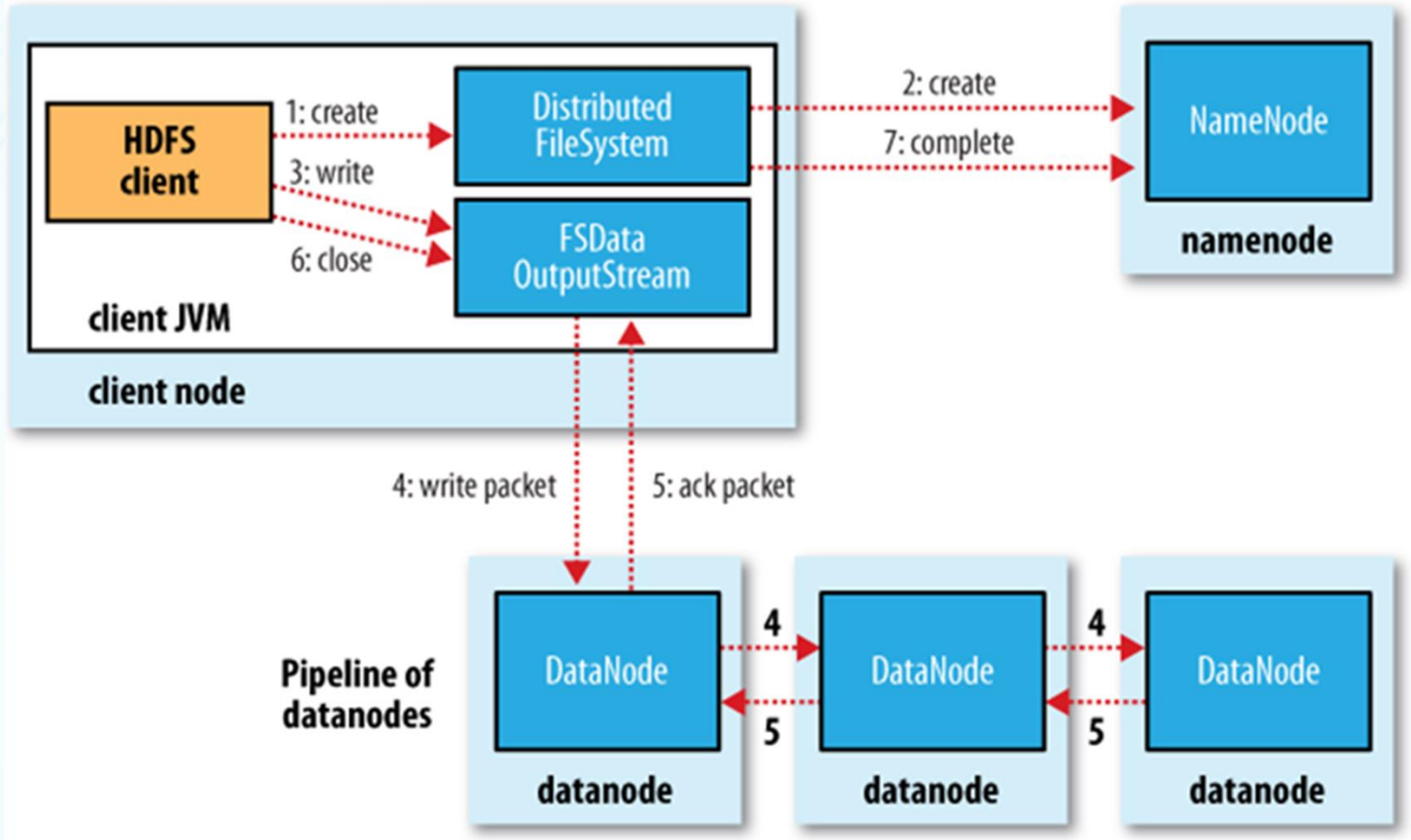
# HFDS Blocks & Replication

- HDFS is a block structured file system.
- Each file is stored on HDFS as blocks of 128MB (default)
- Each block is replicated 3x (default) to deal with Data Node failure



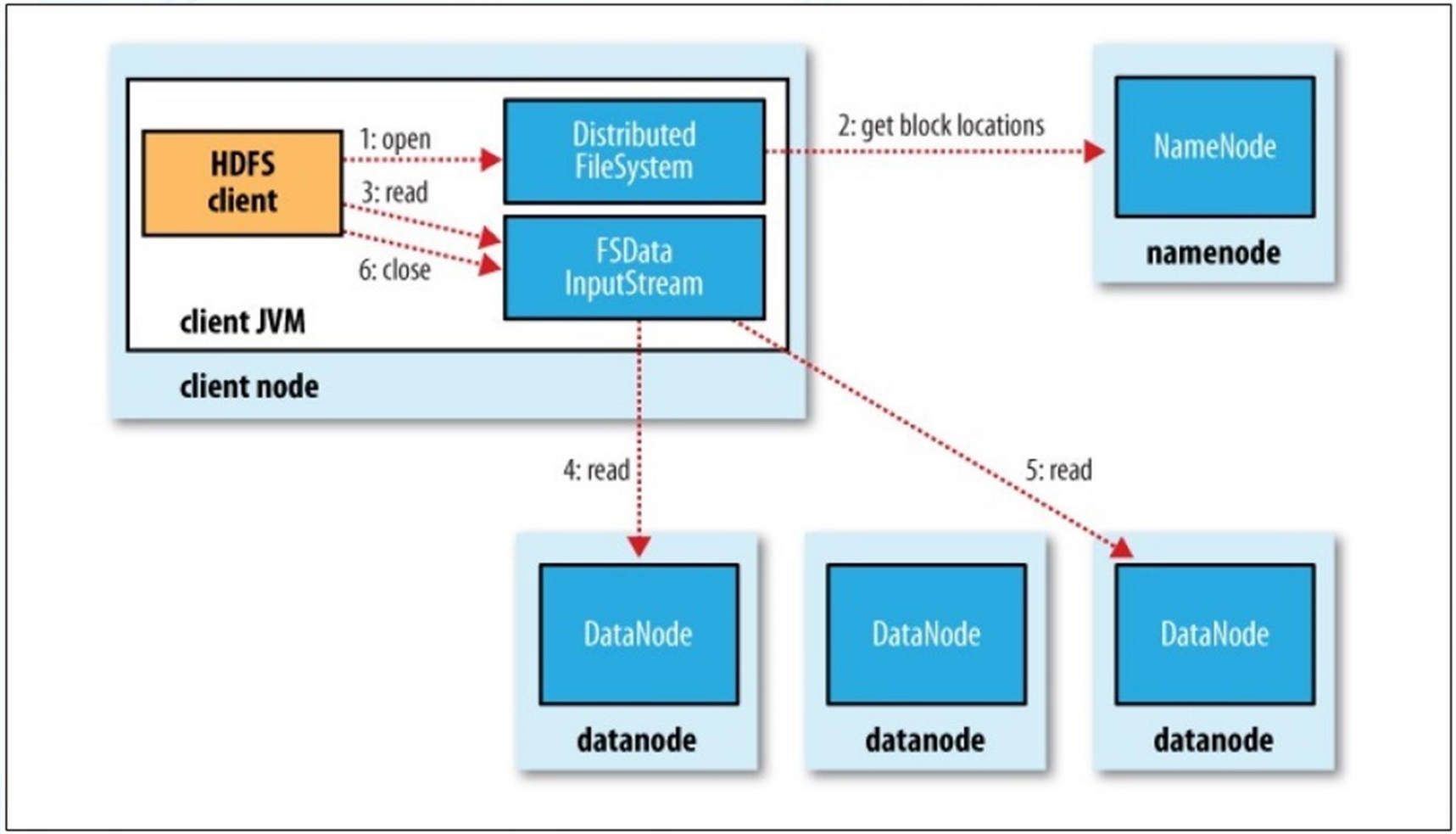
# How are files written to and read from HDFS?

# Anatomy of HDFS File Write

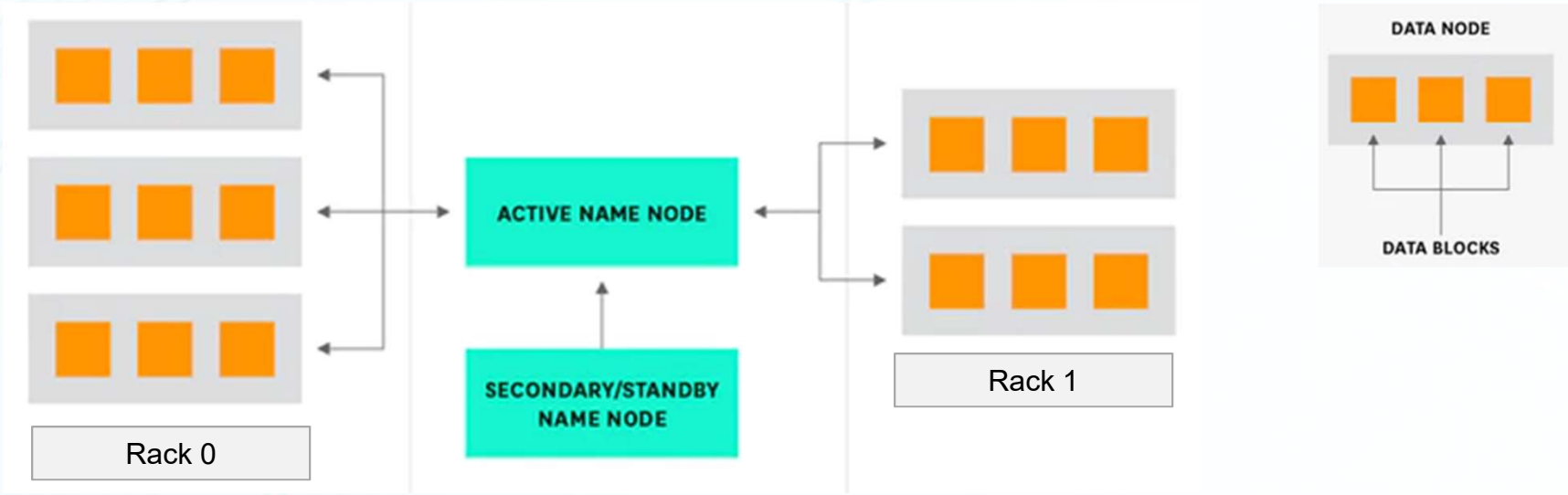



Replications are written sequentially. Blocks are written in parallel.

# Anatomy of HDFS File Read

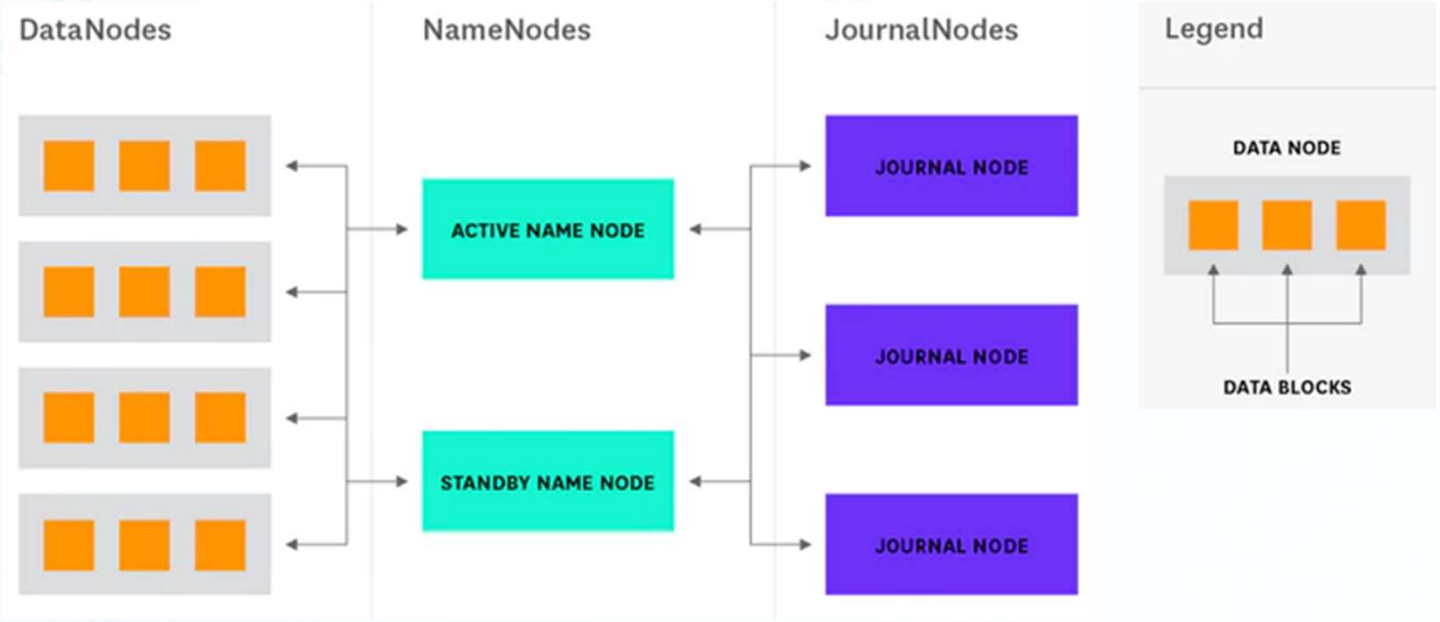


# Hadoop Cluster Architecture with out HA




- In the early versions of Hadoop, NameNode was the ‘single point of failure’.
- SNN does not provide an alternative to NN, but only helps to quickly recover from NN failures. However, if NN is does the cluster experiences a downtime until it is restored.
- With Hadoop 2.0 and Standby NameNodes, a mechanism for true high availability was realized.

# Hadoop Cluster Architecture with HA




- Standby NameNodes provide automatic failover if primary NameNode fails.
- Achieving high availability with Standby NameNodes requires shared storage between the primary and standbys (for the edit log).
- Though there are two options for the necessary shared storage - **NFS** and **Quorum Journal Manager(QJM -** only QJM is considered production-ready.

# Name Node and QJM

- The QJM is a dedicated HDFS implementation, designed for the sole purpose of providing a highly available edit log, and is the recommended choice for most HDFS installations.
- Using QJM to maintain consistency of Active and Standby state requires that both nodes be able to communicate with a group of JournalNodes (JNs).
- When the Active node modifies the namespace, it logs a record of the change to a majority of JournalNodes. The StandbyNode watches the JNs for changes to the edit log and applies them to its own namespace.

# Working with HDFS

# Basic HDFS Commands

## Help

```
$ hadoop fs -help
$ hdfs dfs -help
```

## Basic File Operations

| | |
|---|---|
| `$ hadoop fs -mkdir mydir` | Create directory |
| `$ hadoop fs -rmdir mydir` | Remove a directory |
| `$ hadoop fs -ls` | List files |
| `$ hadoop fs -ls specific_dir` | List files in a dir |
| `$ hadoop fs -ls -R` | Recursive listing |
| `$ hadoop fs -cat myfile` | Read the file content |
| `$ hadoop fs -cp <source> <dist>` | Copy file to a dir with in Hadoop |
| `$ hadoop fs -mv <source> <dist>` | Move a file within hadoop |

# Basic HDFS Commands

| Basic File Operations | |
|---|---|
| `$ hadoop fs -rm myfile` | Delete a file from Hadoop |
| `$ hadoop fs -rm -R mydir` | Delete files recursivly |
| `$ hadoop fs -chmod 777 myfile` | Change permission on a file/dir |
| `$ hadoop fs -chmod -R 777 mydir` | Change permissions recursively |

# Basic HDFS Commands

| Moving files between Linux & HDFS | |
|---|---|
| `$ hadoop fs -copyFromLocal <src> <dist>` | Copy from Linux to HDFS |
| `$ hadoop fs -copyToLocal <src> <dist>` | Copy from HDFS to Linux |
| `$ hadoop fs -get <src> <dist>` | Copy from Linux to HDFS |
| `$ hadoop fs -put <src> <dist>` | Copy from HDFS to Linux |
| `$ hadoop fs -moveFromLocal <src> <dist>` | Move from Linux to HDFS |
| **`$ hadoop distcp -update hdfs://192.168.1.1/xxx hdfs://192.168.1.2/yyy`** | |


TIME FOR QUESTIONS
12
1
2
3
4